REMARKS

Applicant is in receipt of the Office Action mailed March 10, 2008. Reconsideration of the case is earnestly requested in light of the following remarks.

Section 103 Rejections

Claims 1-14, 20-27, and 31 were rejected under 35 U.S.C. 103(a) as being unpatentable over Molinari et al. (U.S. Patent Application Publication No. 2003/0058280 A1, hereinafter "Molinari") in view of Bowman et al. (U.S. Patent No. 6,233,726 B1, hereinafter "Bowman"). Applicant respectfully traverses these rejections.

Amended claim 1 recites as follows:

> 1. (Currently Amended) A computer-readable memory medium storing program instructions executable to:
>
> in source code of a software program, display a first function call written in a text-based programming language that can be compiled into executable code, wherein the first function call takes a first parameter;
>
> programmatically determine one or more valid parameter values for the first parameter of the first function call by invoking software for a measurement device in order to determine one or more resources of the measurement device, wherein each of the one or more valid parameter values represents a respective resource of the one or more resources;
>
> position a cursor on the first function call displayed in the source code in response to user input;
>
> in response to user input requesting to select a parameter value, determine that the cursor is positioned on the first function call and display a graphical user interface for selecting a parameter value for the first parameter of the first function call, wherein the graphical user interface visually indicates the one or more valid parameter values;
>
> receive user input to the graphical user interface to select a first parameter value from the one or more valid parameter values, wherein the first parameter value represents a first resource of the measurement device; and
>
> automatically modify the first function call displayed in the source code of the software program by including the first parameter value in the first function call in response to the user input selecting the first parameter value, wherein automatically including the first parameter value in the first function call aids a user in modifying the first function call to reference the first resource of the measurement device.

The method described in the present application functions to aid a user in modifying a first function call written in a text-based programming language that can be

compiled into executable code. Specifically, the user positions a cursor on the first function call displayed in the source code of the software program. In response to user input requesting to select a parameter value, it is determined that the cursor is positioned on the first function call, and a graphical user interface for selecting a parameter value for a first parameter of the first function call is displayed. In response to user input to the graphical user interface to select a first parameter value, the first function call displayed in the source code of the software program is automatically modified by including the first parameter value in the first function call. This aids a user in modifying the first function call.

The cited references do not teach this subject matter. The principle Molinari reference cited in the claim rejections does not even teach the concept of a first function call written in a text-based programming language that can be compiled into executable code. (As discussed in detail below, Molinari's invention invokes pre-written pieces of compiled code, and does not involve the creation or modification of any source code written in a text-based programming language that can be compiled into executable code.)

The Bowman reference teaches a Parameter Wizard:

> The Parameter Wizard, which operates in conjunction with the Reference Card, guides the user through the construction of function calls. This is particularly useful for complicated functions or for functions with which the user is unfamiliar. The Parameter Wizard helps the user construct a function call by showing what argument values are required. The Parameter Wizard can also store function results in a specified variable.
>
> During use, the Parameter Wizard displays text boxes in a wizard window which lets the user enter variable names, constants, or expressions. The user can use the right mouse button to click the text box and get a list of variables that may be entered. Click on a name from this list to place that name into the text box. The user can also type in entries directly. When the user clicks Finish, the wizard pastes the constructed function call into the code editor window. (*col. 3, lines 17-32*)

Thus, Bowman's Parameter Wizard aids a user in pasting a new function call into the source code. However, the Parameter Wizard does not aid a user in modifying a function call that already exists in the source code. In particular, Bowman, taken either singly or in combination with Molinari, does not teach the recited limitations of:

in source code of a software program, display a first function call written in a text-based programming language that can be compiled into executable code, wherein the first function call takes a first parameter;

...

position a cursor on the first function call displayed in the source code in response to user input;

in response to user input requesting to select a parameter value, determine that the cursor is positioned on the first function call and display a graphical user interface for selecting a parameter value for the first parameter of the first function call

...

automatically modify the first function call displayed in the source code of the software program by including the first parameter value in the first function call in response to the user input selecting the first parameter value, wherein automatically including the first parameter value in the first function call aids a user in modifying the first function call to reference the first resource of the measurement device.

Thus, Applicant respectfully submits that claim 1 is patentably distinct over the cited art for at least this first reason.

Applicant also respectfully submits that Molinari and Bowman do not teach the recited limitations of:

programmatically determine one or more valid parameter values for the first parameter of the first function call by invoking an application programming interface (API) of manager software for a measurement device in order to determine one or more resources of the measurement device, wherein each of the one or more valid parameter values represents a respective resource of the one or more resources;

Molinari teaches a system that enables a user to develop a measurement application by interacting with graphical panels. Molinari teaches displaying DAQ channels, e.g., as shown in FIG. 13. However, Molinari does not teach how these DAQ channels are determined. In particular, Molinari does not teach that the DAQ channels are determined by invoking an application programming interface (API) of manager software for the DAQ device (measurement device). Applicant respectfully reminds the Examiner that the prior art must teach the limitations specifically recited in the claim. Applicant notes Molinari's invention may determine the DAQ channels without invoking an application programming interface (API) of manager software for the DAQ device.

For example, Molinari's invention may simply read the DAQ channels from a file that specifies the configuration of the DAQ device. Molinari nowhere teaches the specifically recited limitations of, "invoking an application programming interface (API) of manager software for a measurement device in order to determine one or more resources of the measurement device".

Thus, Applicant respectfully submits that claim 1 is also patentably distinct over the cited art for at least this second reason.

Furthermore, Applicant respectfully submits that there is no motivation for incorporating Bowman's teaching into Molinari's invention as proposed by the Examiner. Molinari teaches that:

> The development of instrumentation systems software by professionals using textual programming languages such as C++ is very time consuming and tedious, and it typically results in the production of a program consisting of many pages of so-called source code, written in a computer language that is virtually unreadable by non-programmers, and which thus cannot be modified by the typical users of such programs. (*paragraph 0014*)

Thus, Molinari teaches throughout the disclosure that the use of text-based programming languages is avoided so that a user with no software programming training or experience is able to design a measurement application. See, for example, paragraph 0019:

> The present invention is a graphical measurement application development software system. That is to say, the invention is a software application that presents the user with a graphical desktop, and sets of software "tools", for developing a problem-specific test or measurement application by using graphical programming techniques such as mouse manipulations of icons and of drop down lists of selectable options. Using the tools provided by the invention, a user having essentially no software programming training or experience is enabled to design and construct a working, problem-specific measurement solution (*paragraph 0019*)

Instead of using of text-based programming language, the user develops a set of panels and interconnects the panels to develop the application. See, for example, paragraphs 0025 and 0026:

The procedure employed by the user in developing an application is generally as follows: upon opening the software of the invention, the user is presented with a "design mode" desktop, having the general appearance of a graphical desktop containing a large workspace. By selections from menu lists, or the "drag and drop" of selected panel icons presented in "flying tool windows", the user places on the design workspace a selection of "panels", chosen to represent the several instrument components that will need to be combined to form an "instrument" of the kind required by the user's intended application. (*paragraph 0025*)

There is associated with each panel a property page, readily accessible to the user, that serves as the main tool for configuring the particular component and adapting it to suit the user's requirements, and to connect that instrument component panel to other component panels. (*paragraph 0026*)

Furthermore, Molinari's invention requires no compilation or interpretation of code:

upon a user's development of a specific application, through the selection and configuration of specific "panels" corresponding to software aspects, the execution of the developed application may therefore be implemented, with no compilation or interpretation of code, by the execution of a simple textual script file (*paragraph 0032*)

See also paragraph 0034:

By contrast, user programs developed using the present invention require no compilation or interpretation, because the machine-readable code required to execute all program functions is pre-compiled and archived. (*paragraph 0034*)

Molinari clearly teaches that avoiding the use of source code written in a text-based programming language is an important aspect of the invention. In particular, see paragraph 0036:

An important advantage of the present invention, and its use of "aspects" as software objects directly associated with executable code segments maintained in libraries, is that a measurement application created by a user of the invention may be fully represented in a brief text description of the chosen and configured aspects, and of their interconnections, in lieu of pages of source code. In particular, a user in creating an application program using the present invention, by using graphical tools to select and configure "panels" that represent software aspects, is simultaneously caused to create such a text file as the run time and distributable representation of the user's application. In embodiments described in the present specification, the format selected to represent and comprise user-designed programs is termed Aspect Interaction Language (AIL), an expression selected to reflect the nature of AIL as a form of program representation that is

purely textual and descriptive, and that is not a programming language subject to being compiled into machine-executable code. (*paragraph 0036*)

See also paragraphs 0099 and 0100:

The actions of the user in selecting instrument component panels, and configuring their properties, serve to define the content and properties of the user's intended application program. These actions by the user also operate to create a textual file, called in this description an Aspect Interaction Language (AIL) file, that contains a description of the selected, created and defined software "aspects", including, for each aspect, a description of its properties and connections. As the user adds more panels, selects more properties, and effects connections between panels, the AIL file expands into a complete textual description of the user's program.

Using AIL description language, the file embodying the user's application comprises only a relatively few, human-readable lines that set forth identifications and descriptions of the aspects used in the application, and their interconnections and interactions, in lieu of hundreds of lines of source code needing to be parsed and compiled or interpreted. Using the AIL approach, no compiling or interpretation is needed.

Thus, Molinari teaches very clearly throughout the disclosure that the invention does not involve the creation or editing of source code written in a text-based programming language that can be compiled into executable code. Instead, Molinari's invention simply invokes pre-written pieces of compiled code:

User programs created using the present invention are accordingly always optimized, and ready-to-run immediately upon completion of program design. Each aspect employed in the user's program is comprised of pre-written pieces of compiled code. The user can generally modify at will the behavior or properties of a selected aspect, but in each such instance the code corresponding to the modification is already compiled, and standing by for possible use. (*paragraph 0103*)

For the user of a graphical programming environment according to the present invention, the system is remarkably fast in operation, because the user develops a program using as tools only basic descriptions of aspects, which in turn represent existing and optimized executable code segments. Thus no compilation or interpretation of code is necessary to execute a program designed by the user. (*paragraph 0105*)

It is very clear that Molinari's invention does not use source code written in a text-based programming language that can be compiled into executable code. In sharp

contrast, Bowman teaches a Parameter Wizard that aids a user in constructing source code written in a text-based programming language that can be compiled into executable code. There is no reason to insert a function call into source code as taught by Bowman because in Molinari's invention there is no source code to be compiled. Modifying Molinari's system to use source code written in a text-based programming language would be a clear violation of several principles of operation of Molinari's invention.

In response to these arguments the Examiner states on p. 39 of the current Office Action:

> Molinari does not teach away from the combination with Bowman. Molinari discloses an interface for the user to use in lieu of personally writing the source code. However, the program writes the underlying source code for the user. The program takes the user's inputs and turns it into source code in the form of the AIL file.

The Examiner's assertion that the AIL file constitutes source code directly contradicts Molinari's disclosure. As discussed above, at paragraph 0036 Molinari clearly states that the AIL file is a "brief text description of the chosen and configured aspects, and of their interconnections, *in lieu of pages of source code*". Thus Molinari very clearly states that the AIL file is not source code.

The Examiner also states that:

> In other words, the AIL file is turned into an executable (i.e., compiled) and the abstract representation of existing, executable code segments by the AIL file, is in essence, source code. Therefore, Molinari does not teach away from the combination with Bowman.

Applicant disagrees. Molinari nowhere teaches that the AIL file is compiled or turned into an executable, as asserted by the Examiner. In fact, Molinari emphasizes numerous times throughout the disclosure that the invention does not require any compilation. For example, as discussed above, paragraph 0034 teaches:

> By contrast, user programs developed using the present invention require no compilation or interpretation, because the machine-readable code required to execute all program functions is pre-compiled and archived. (*paragraph 0034*)

Molinari's invention simply invokes pre-written pieces of compiled code, as discussed above. There is no reason to insert a function call into source code as taught by Bowman because in Molinari's invention there is no source code to be compiled.

Applicant thus respectfully submits that there is no motivation to combine Bowman with Molinari, and thus, claim 1 is also patentably distinct over the cited art for at least this third reason.

Inasmuch as the independent claims 24, 26, and 27 recite similar limitations as those of claim 1, Applicant respectfully submits that these claims are also patentably distinct over Molinari and Bowman, for reasons similar to those discussed above.

As for the independent claim 31, the claim recites in pertinent part:

> display a block diagram of a graphical program, wherein the block diagram includes a plurality of interconnected nodes visually indicating functionality of the graphical program, wherein the block diagram can be compiled into executable code, wherein the plurality of interconnected nodes includes a first node that takes a first input parameter;

The cited references do not teach these limitations in combination with the other limitations recited in claim 31. The Examiner has interpreted the recited graphical program with FIG. 1 of Molinari. However, FIG. 1 simply illustrates a graphical user interface of a measurement application developed using Molinari's system. A graphical user interface is not at all the same as a graphical program. The graphical user interface of FIG. 1 simply provides an interface to executable code. The graphical user interface itself cannot be compiled into executable code, as recited in claim 31.

Furthermore, with respect to the limitation of compiling the block diagram into executable code, the Examiner cites various portions of Molinari's teaching regarding the AIL file. However, as discussed above, Molinari teaches that developing a user application does not involve any compilation.

Furthermore, claim 31 also recites:

> automatically configure the first node with the first parameter value in response to the user input selecting the first parameter value, wherein automatically configuring the first node with the first parameter value comprises

automatically updating the displayed block diagram to visually indicate that the first node receives the first parameter value as input.

With respect to these limitations, the Examiner cites Bowman's teaching of changing C++ code, i.e., a text-based source code. However, claim 31 does not recite anything about changing C++ code or other text-based source code, but instead recites, "automatically configure the first node with the first parameter value in response to the user input selecting the first parameter value, wherein automatically configuring the first node with the first parameter value comprises automatically updating the displayed block diagram to visually indicate that the first node receives the first parameter value as input." The references do not teach these limitations. Changing text-based source code is not at all the same as changing a block diagram of a graphical program.

Applicant thus respectfully submits that claim 31 is patentably distinct over the cited art for at least the reasons set forth above.

Since the independent claims have been shown to be patentably distinct over Molinari and Bowman, Applicant submits that the dependent claims are also patentably distinct, for at least this reason. Applicant also submits that numerous ones of the dependent claims recite further distinctions over Molinari and Bowman. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-77600/JCH.

Also filed herewith are the following items:

☒ Request for Continued Examination

☐ Terminal Disclaimer

☐ Power of Attorney By Assignee and Revocation of Previous Powers

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 2008-06-10 JCH/JLB